AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method for sending a request from a client computer to a server computer, wherein the client computer is connected to the server computer via a network, wherein the server computer runs server-side code associated with a client/server application, wherein the client computer runs client-side code associated with the client/server application, the method comprising:

the client-side code registering a request entry with a client-side scheduler while the client computer is unable to establish a connection to ~~the network~~ the server computer over the network ~~is unavailable~~, wherein the request entry represents the request to be sent from the client computer to the server computer;

the client-side scheduler storing the request entry until a connection ~~to the network is available~~ from the client computer to the server computer can be established for use by the client-side code;

the client-side scheduler detecting when a connection from the client computer to the server computer is established;

the client-side scheduler notifying the client-side code that the connection from the client computer to the server computer over the network ~~is available~~ is established;

the client-side code using the connection to the ~~network~~ server computer to send the request to the server computer.

2. (Currently Amended) The method of claim 1,

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code informing the client-side scheduler of a callback routine to associate with the request entry;

wherein said client-side scheduler storing the request entry comprises the client-side scheduler storing information identifying the callback routine;

wherein said client-side scheduler notifying the client-side code that the ~~network~~ connection from the client computer to the server computer is established ~~available~~ comprises the client-side scheduler invoking the callback routine.

3. (Currently Amended) The method of claim 1,

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code storing request context information for the request entry;

~~wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code storing request context information for the request entry;~~

wherein said client-side code using the ~~network~~ connection from the client computer to the server computer to send the request to the server computer comprises the client-side code retrieving the request context information to formulate the appropriate request to send to the server computer.

4. (Currently Amended) The method of claim 3,

wherein said client-side code storing request context information for the request entry comprises the client-side code storing the request context information in a ~~data base~~ database.

5. (Currently Amended) The method of claim 4,

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code informing the client-side scheduler of a database key identifier for the request context information stored in the ~~data base~~ database;

wherein said client-side scheduler storing the request entry comprises the client-side scheduler storing the database key identifier;

wherein said client-side scheduler notifying the client-side code that the ~~network~~ connection from the client computer to the server computer is established ~~available~~ comprises the client-side scheduler passing the database key identifier to the client-side code;

wherein said client-side code retrieving the request context information for the request entry comprises the client-side code using the database key identifier to retrieve the request context information from the database.

6. (Currently Amended) The method of claim 1,

wherein the that the client-side code needs to send to the server computer requires exactly-once semantics;

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code informing the client-side scheduler that the request requires exactly-once semantics;

wherein said client-side scheduler storing the request entry comprises the client-side scheduler storing information specifying that the request requires exactly-once semantics;

wherein, in the case of an error during said client-side code using the ~~network~~ connection from the client computer to the server computer to send the request to the server computer, the method includes ~~means for~~ providing error recovery for the request~~, wherein said means for providing error recovery ensure~~ and ensuring that the server computer does not perform the request more than once.

7. (Currently Amended) The method of claim 6,

wherein said error during the client-side code using the ~~network~~ connection from the client computer to the server computer to send the request to the server computer comprises terminating the network connection from the client computer to the server computer ~~being terminated~~ before the response to the request is received from the server computer;

wherein ~~said means for~~ providing error recovery for the request comprises:

~~The~~ the client-side code informing the client-side scheduler that the request failed;

~~The~~ the client-side scheduler marking the request entry corresponding to the request with a status indicating that the request needs error recovery, in response to being informed by the client-side code that the request failed;

~~A network~~ re-establishing a connection from the client computer to the server computer ~~being re-established~~;

~~The~~ the client-side scheduler using the ~~network~~ connection from the client computer to the server computer to query the server computer in order to determine whether the request was successfully completed;

~~The~~ the client-side scheduler informing the client-side code that the request was successfully completed if the client-side scheduler determines from the query to the server computer that the request was successfully completed;

~~The~~ the client-side scheduler marking the request entry corresponding to the request with a status indicating that the request should be re-sent if the client-side scheduler determines from

the query to the server computer that the request was not successfully completed.

8. (Original) The method of claim 7,

wherein the request sent by the client-side code to the server computer includes a job ID for the request;

wherein, in response to successfully performing the request, the server computer is operable to log the job ID for the request;

wherein said client-side scheduler querying the server computer in order to determine whether the request was successfully completed comprises the client-side scheduler querying the server computer in order to determine whether the server computer logged the job ID for the request.

9. (Original) The method of claim 1,

wherein the client-side code maintains a client-side cache;

wherein, in response to determining a need for a particular information, the client-side code is operable to check the client-side cache for the needed information;

wherein, if the client-side cache comprises the needed information, the client-side code is operable to retrieve the needed information from the client-side cache;

wherein, if the client-side cache does not comprise the needed information, the client-side code is operable to register a request entry with the client-side scheduler, wherein the request entry represents a request to retrieve the needed information from the server computer.

10. (Original) The method of claim 9,

wherein the client-side cache is implemented using a database.

11. (Currently Amended) The method of claim 9,

wherein the server-side code maintains a server-side database;

wherein the client-side code is operable to register a synchronization entry with the client-side scheduler;

wherein the synchronization entry represents a synchronization request to be sent by the client-side code to the server computer;

wherein the client-side scheduler stores the synchronization entry until a ~~network~~ connection from the client computer to the server computer is established ~~available for use by the client-side code~~;

wherein the client-side scheduler notifies the client-side code that the ~~network~~ connection from the client computer to the server computer is established ~~available~~;

wherein the client-side code is operable to send the synchronization request to the server computer and use the synchronization request results returned by the server computer to synchronize at least a portion of the client-side cache with the server-side database.

12. (Original) The method of claim 11,

wherein said synchronization request sent to the server computer comprises a list of item IDS and modification times;

13. (Currently Amended) The method of claim 11,

wherein said client-side code registering a synchronization entry with the client-side scheduler comprises the client-side code informing the client-side scheduler of a synchronization callback routine to associate with the synchronization entry;

wherein the client-side scheduler stores information identifying the synchronization call routine;

wherein the client-side scheduler invokes the synchronization callback routine when a ~~network~~ connection from the client computer to the server computer is established ~~available~~ for use by the synchronization callback routine.

14. (Currently Amended) The method of claim 13,

wherein said client-side code registering a synchronization entry with the client-side scheduler further comprises the client-side code informing the client-side scheduler of a time to invoke the synchronization callback routine;

wherein the client-side scheduler stores information identifying the time to invoke the synchronization callback routine;

wherein the client-side scheduler invokes the synchronization callback routine when the specified invocation time occurs and a ~~network~~ connection from the client computer to the server

computer is established ~~available~~ for use by the synchronization callback routine.

15. (Currently Amended) The method of claim 14,

wherein, upon completing the synchronization of at least a portion of the client-side cache with the server-side database, the synchronization callback routine informs the scheduler of a time to re-invoke the synchronization callback routine;

wherein the scheduler stores information identifying the time to re-invoke the synchronization callback routine;

wherein the scheduler re-invokes the synchronization callback routine when the specified re-invocation time occurs and a ~~network~~ connection from the client computer to the server computer is established ~~available~~ for use by the synchronization callback routine.

16. (Original) The method of claim 15,

wherein the re-invocation time specified by the synchronization callback routine is calculated based on the amount of synchronization that the synchronization callback routine performed.

17. (Currently Amended) The method of claim 11,

wherein a request entry and a synchronization entry are simultaneously registered with the client-side scheduler;

wherein the client-side scheduler prioritizes the request entry over the synchronization entry with regard to notifying the client-side code that a ~~network~~ connection from the client computer to the server computer is established ~~available~~.

18. (Original) The method of claim 1,

wherein the method comprises error recovery means in the event of abnormal termination of the client-side code;

wherein said error recovery means comprise ensuring that upon restart of the client-side code, any requests requiring exactly-once semantics that were in progress when the abnormal termination of the client-side code occurred are not performed multiple times.

19. (Original) The method of claim 1,

wherein the server side of the client/server application comprises a first server computer;

wherein requests sent by the client-side code are received by the first server computer and passed to a second server computer;

wherein the second server computer carries out the request.

20. (Original) The method of claim 1,

wherein the client-side scheduler is an independent software component.

21. (Original) The method of claim 20,

wherein the client-side scheduler component is a component constructed according to a software component specification from the group consisting of: COM, CORBA, JavaBeans.

22. (Original) The method of claim 1,

wherein the client-side scheduler is not an independent software component, but rather is tightly integrated with the client-side code.

23. (Original) The method of claim 1,

wherein the client-side scheduler executes on the client computer.

24. (Original) The method of claim 1,

wherein the client-side scheduler executes on a computer connected to the client computer via a local area network (LAN).

25. (Original) The method of claim 1, wherein the client/server application is an Internet application; wherein the network is the Internet

26. (Original) The method of claim 25,

wherein the Internet application is a healthcare application.

27. (Original) The method of claim 26,

wherein the healthcare application enables healthcare workers to file health insurance claims while the client computer is disconnected from the Internet.

28. (Original) The method of claim 25,

wherein the request sent by the client-side code references a resource accessible over the Internet via a uniform resource locator (URL).

29. (Original) The method of claim 28,

wherein the client-side code interfaces with the Web browser code base in order to send the request over the Internet.

30. (Original) The method of claim 28,

wherein the client-side code uses the Web browser control as an embedded component.

31. (Currently Amended) A system for enabling a client/server application, the system comprising:

a server computer, wherein the server computer runs server-side code associated with the application;

a client computer connected to the server computer via a network, wherein the client computer runs client-side code associated with the application; and

a client-side scheduler, wherein the client-side code is operable to interface with the client-side scheduler in order to manage requests to the server computer;

wherein the client-side code is executable to register a request entry with the client-side scheduler while the client computer is unable to establish a connection to ~~the network~~ the server computer over the network ~~is unavailable~~, wherein the request entry represents a request that the client-side code needs to send to the server computer;

wherein the client-side scheduler is executable to store the request entry until ~~a connection to the network is available~~ a connection from the client computer to the server computer can be established for use by the client-side code;

wherein the client-side scheduler is executable to detect when a connection from the client computer to the server computer is established;

wherein the client-side scheduler is executable to notify the client-side code that the connection from the client computer to the server computer over the network ~~is available~~ is established;

wherein the client-side code is executable to use the connection to the ~~network~~ server computer to send the request to the server computer.

32. (Currently Amended) The system of claim 31,

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code informing the client-side scheduler of a callback routine to associate with the request entry;

wherein said client-side scheduler storing the request entry comprises the client-side scheduler storing information identifying the callback routine;

wherein said client-side scheduler notifying the client-side code that the ~~network~~ connection from the client computer to the server computer is established ~~available~~ comprises the client-side scheduler invoking the callback routine.

33. (Currently Amended) The system of claim 31,

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code storing request context information for the request entry;

~~wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code storing request context information for the request entry;~~

wherein said client-side code using the ~~network~~ established connection from the client computer to the server computer to send the request to the server computer comprises the client-side code retrieving the request context information to formulate the appropriate request to send to the server computer.

34. (Original) The system of claim 33,

wherein said client-side code storing request context information for the request entry comprises the client-side code storing the request context information in a database.

35. (Currently Amended) The system of claim 34,

wherein said client-side code registering a request entry with the scheduler comprises the client-side code informing the scheduler of a database key identifier for the request context information stored in the database;

wherein said scheduler storing the request entry comprises the scheduler storing the database key identifier;

wherein said scheduler notifying the client-side code that the ~~network~~ connection from the client computer to the server computer is established ~~available~~ comprises the scheduler passing the database key identifier to the client-side code;

wherein said client-side code retrieving the request context information for the request entry comprises the client-side code using the database key identifier to retrieve the request context information from the database.

36. (Currently Amended) The system of claim 31,

wherein the that the client-side code needs to send to the server computer requires exactly-once semantics;

wherein said client-side code registering a request entry with the client-side scheduler comprises the client-side code informing the client-side scheduler that the request requires exactly-once semantics;

wherein said client-side scheduler storing the request entry comprises the client-side scheduler storing information specifying that the request requires exactly-once semantics;

wherein, in the case of an error during said client-side code using the ~~network~~ established connection from the client computer to the server computer to send the request to the server computer, the method includes means for providing error recovery for the request, wherein said means for providing error recovery ensure that the server computer does not perform the request more than once.

37. (Currently Amended) The system of claim 36,

wherein said error during the client-side code using the ~~network~~ established connection from the client computer to the server computer to send the request to the server computer comprises the network connection between the client computer and the server computer being terminated before the response to the request is received from the server computer;

wherein said means for providing error recovery for the request comprise:

the client-side code informing the client-side scheduler that the request failed;

the client-side scheduler marking the request entry corresponding to the request with a status indicating that the request needs error recovery, in response to being informed by the client-side code that the request failed;

a ~~network~~ connection from the client computer to the server computer being re-established;

the client-side scheduler using the ~~network~~ re-established connection from the client computer to the server computer to query the server computer in order to determine whether the request was successfully completed;

the client-side scheduler informing the client-side code that the request was successfully completed if the client-side scheduler determines from the query to the server computer that the request was successfully completed;

the client-side scheduler marking the request entry corresponding to the request with a status indicating that the request should be re-sent if the client-side scheduler determines from the query to the server computer that the request was not successfully completed.

38. (Original) The system of claim 37,

wherein the request sent by the client-side code to the server computer includes a job ID for the request;

wherein, in response to successfully performing the request, the server computer is operable to log the job ID for the request;

wherein said client-side scheduler querying the server computer in order to determine whether the request was successfully completed comprises the client-side scheduler querying the server computer in order to determine whether the server computer logged the job ID for the

request.

39. (Original) The system of claim 31,

wherein the client-side code maintains a client-side cache;

wherein, in response to determining a need for a particular information, the client-side code is operable to check the client-side cache for the needed information;

wherein, if the client-side cache comprises the needed information, the client-side code is operable to retrieve the needed information from the client-side cache;

wherein, if the client-side cache does not comprise the needed information, the client-side code is operable to register a request entry with the client-side scheduler, wherein the request entry represents a request to retrieve the needed information from the server computer.

40. (Original) The system of claim 39,

wherein the client-side cache is implemented using a database.

41. (Currently Amended) The system of claim 39, further comprising a server-side database;

wherein the server-side code maintains a server-side database;

wherein the client-side code is operable to register a synchronization entry with the client-side scheduler;

wherein the synchronization entry represents a synchronization request to be sent by the client-side code to the server computer;

wherein the client-side scheduler stores the synchronization entry until a ~~network~~ connection from the client computer to the server computer can be established ~~is available~~ for use by the client-side code;

wherein the client-side scheduler notifies the client-side code that the ~~network~~ connection from the client computer to the server computer is ~~available~~ established;

wherein the client-side code is operable to send the synchronization request to the server computer using the established connection and use the synchronization request results returned by the server computer to synchronize at least a portion of the client-side cache with the server-side database.

42. (Original) The system of claim 41,

wherein said synchronization request sent to the server computer comprises a list of item IDS and modification times;

wherein, in response to receiving the synchronization request, the server computer is operable to return a list of new items to add to the client-side cache, a list of items to delete form the client-side cache, and a list of items to update in the client-side cache.

43. (Currently Amended) The system of claim 41,

wherein said client-side code registering a synchronization entry with the client-side scheduler comprises the client-side code informing the client-side scheduler of a synchronization callback routine to associate with the synchronization entry;

wherein the client-side scheduler stores information identifying the synchronization call routine;

wherein the client-side scheduler invokes the synchronization callback routine when a ~~network~~ connection from the client computer to the server computer is ~~available~~ established for use by the synchronization callback routine.

44. (Currently Amended) The system of claim 43,

wherein said client-side code registering a synchronization entry with the client-side scheduler further comprises the client-side code informing the client-side scheduler of a time to invoke the synchronization callback routine;

wherein the client-side scheduler stores information identifying the time to invoke the synchronization callback routine;

wherein the client-side scheduler invokes the synchronization callback routine when the specified invocation time occurs and a ~~network~~ connection from the client computer to the server computer is ~~available~~ established for use by the synchronization callback routine.

45. (Currently Amended) The system of claim 44,

wherein, upon completing the synchronization of at least a portion of the client-side cache with the server-side database, the synchronization callback routine informs the client-side scheduler of a time to re-invoke the synchronization callback routine;

wherein the client-side scheduler stores information identifying the time to re-invoke the synchronization callback routine;

wherein the client-side scheduler re-invokes the synchronization callback routine when the specified re-invocation time occurs and a ~~network~~ connection from the client computer to the server computer is ~~available~~ established for use by the synchronization callback routine.

46. (Original) The system of claim 45,

wherein the re-invocation time specified by the synchronization callback routine is calculated based on the amount of synchronization that the synchronization callback routine performed.

47. (Currently Amended) The system of claim 41,

wherein a request entry and a synchronization entry are simultaneously registered with the client-side scheduler;

wherein the client-side scheduler prioritizes the request entry over the synchronization entry with regard to notifying the client-side code that a ~~network~~ connection from the client computer to the server computer is ~~available~~ established.

48. (Original) The system of claim 31,

wherein the method comprises error recovery means in the event of abnormal termination of the client-side code;

wherein said error recovery means comprise ensuring that upon restart of the client-side code, any requests requiring exactly-once semantics that were in progress when the abnormal termination of the client-side code occurred are not performed multiple times.

49. (Original) The system of claim 31,

wherein the server side of the client/server application comprises a first server computer;

wherein requests sent by the client-side code are received by the first server computer and passed to a second server computer;

wherein the second server computer carries out the request.

50. (Original) The system of claim 31,

wherein the client-side scheduler is an independent software component.

51. (Original) The system of claim 50,

wherein the client-side scheduler component is a component constructed according to a software component specification from the group consisting of: COM, CORBA, JavaBeans.

52. (Original) The system of claim 31,

wherein the client-side scheduler is not an independent software component, but rather is tightly integrated with the client-side code.

53. (Original) The system of claim 31,

wherein the client-side scheduler executes on the client computer.

54. (Original) The system of claim 31,

wherein the client-side scheduler executes on a computer connected to the client computer via a local area network (LAN).

55. (Original) The system of claim 31,

wherein the client/server application is an Internet application; wherein the network is the Internet.

56. (Original) The system of claim 55,

wherein in Internet application is a healthcare application.

57. (Original) The system of claim 56,

wherein the healthcare application enables healthcare workers to file health insurance claims while the client computer is disconnected from the Internet.

Claims 58-62 (Cancelled)